

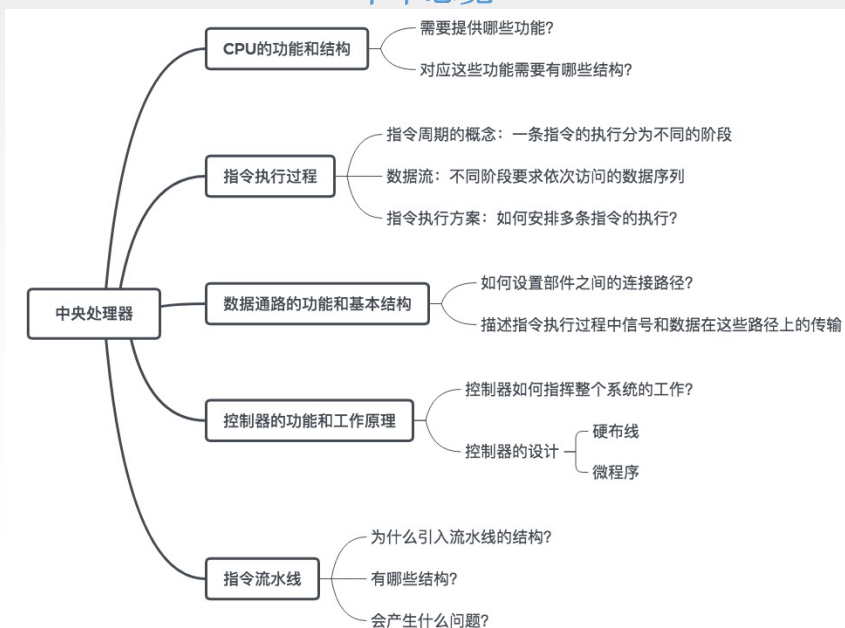
本节内容

微程序控制器的
基本原理

王道考研/CSKAOYAN.COM

1


本章总览



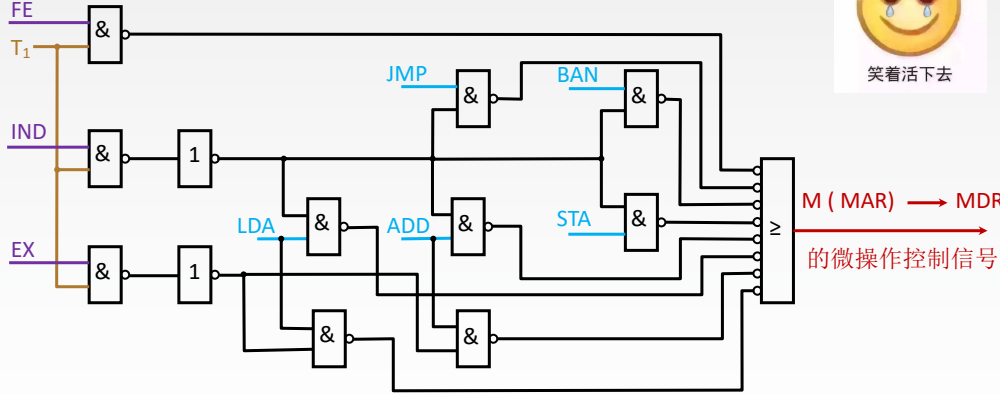
王道考研/CSKAOYAN.COM

2

硬布线控制器的设计思路



笑着活下去



M (MAR) → MDR
的微操作控制信号

硬布线控制器：微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生

时序信息包含机器周期、节拍

王道考研/CSKAOYAN.COM


3

微程序控制器的设计思路

采用“存储程序”的思想，CPU 出厂前将所有指令的“微程序”存入“控制器存储器”中

高级语言代码
高级语言代码

指令1
指令2
指令3
指令4
指令5
指令6



禁止套娃

取指周期 (FE=1)

间指周期 (IND=1)

执行周期 (EX=1)

中断周期 (INT=1)

T ₀ : 微操作1、微操作2	微指令a: 完成微操作1、2
T ₁ : 微操作3	微指令b: 完成微操作3
T ₂ : 微操作4	微指令c: 完成微操作4
T ₀ : 微操作5、微操作2	微指令d: 完成微操作5、2
T ₁ : 微操作6	微指令e: 完成微操作6
T ₂ : 微操作7	微指令f: 完成微操作7
T ₀ : 微操作7	微指令g: 完成微操作7
T ₁ : 微操作8	微指令h: 完成微操作8
T ₂ : 微操作9、微操作6	微指令i: 完成微操作9、6
T ₀ : 微操作5、微操作2	微指令j: 完成微操作5、2
T ₁ : 微操作10	微指令k: 完成微操作10
T ₂ : 微操作11	微指令l: 完成微操作11

程序：由指令序列组成
微程序：由微指令序列组成，每一种指令对应一个微程序

指令是对程序执行步骤的描述
微指令是对指令执行步骤的描述
指令是对微指令功能的“封装”

微指令基本格式

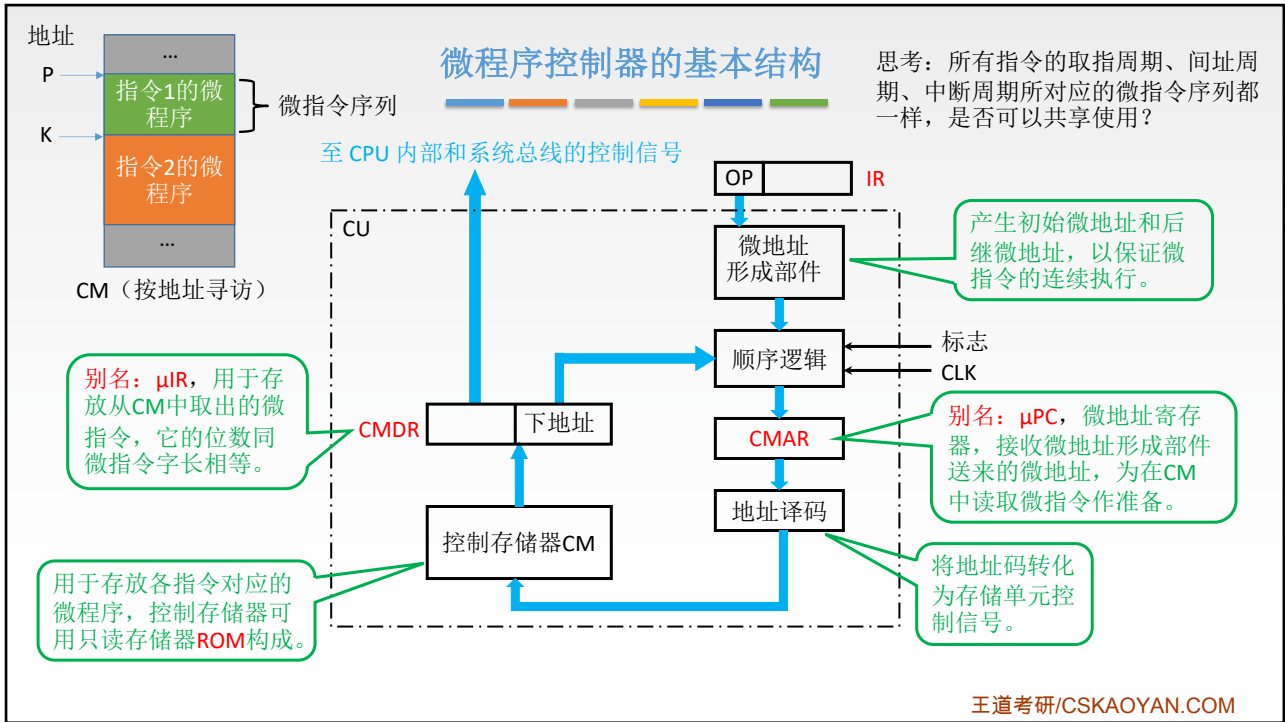
← 操作控制 →
← 顺序控制 →

指明下一条微指令的地址

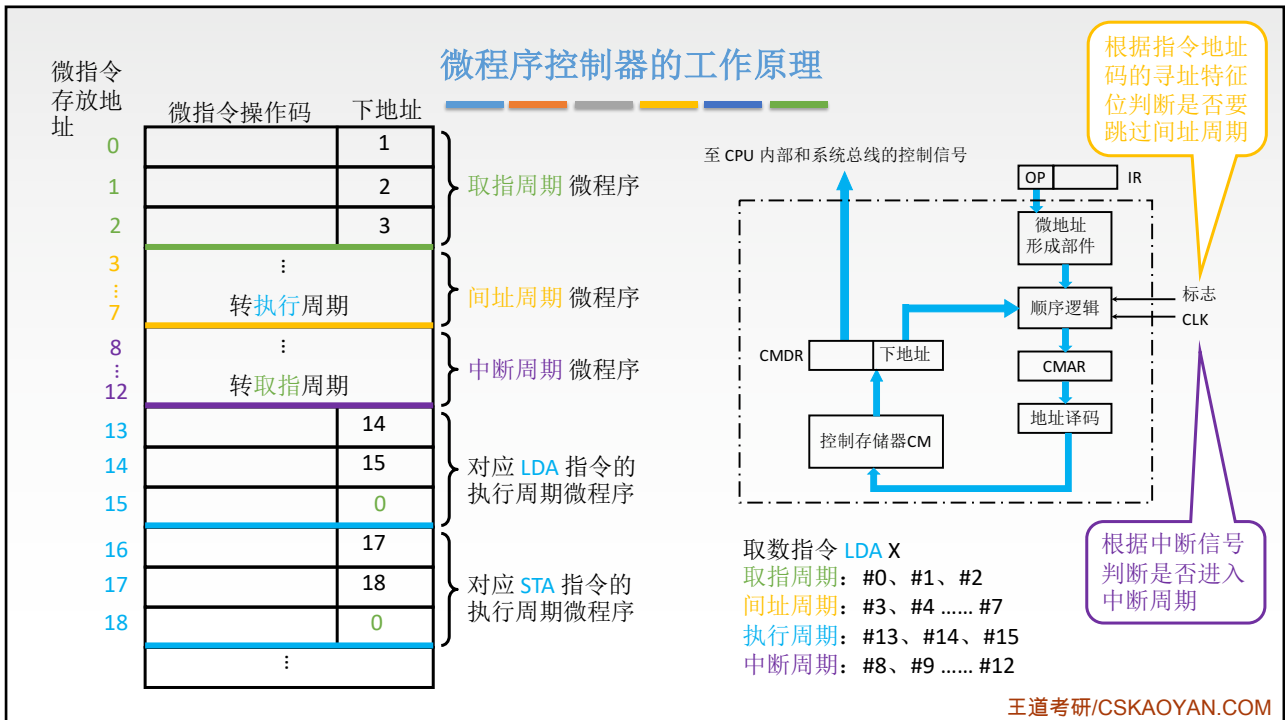
微命令与微操作一一对应
微指令中可能包含多个微命令

王道考研/CSKAOYAN.COM

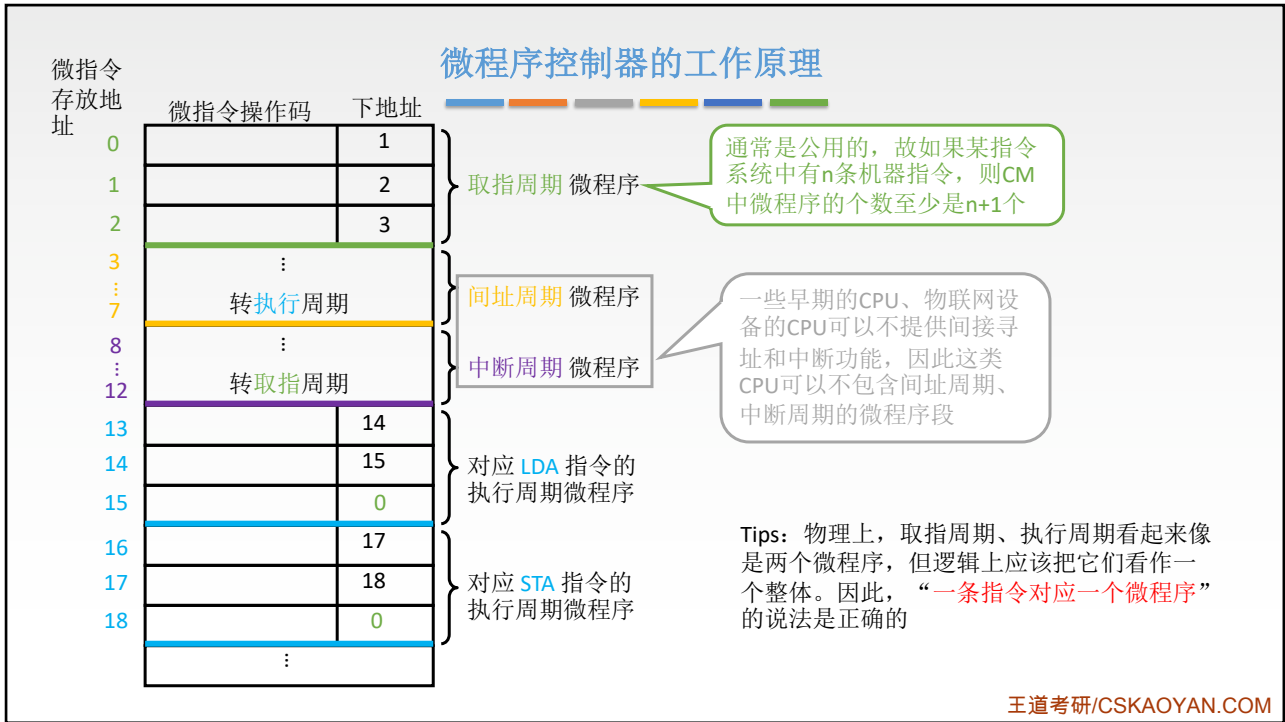
4



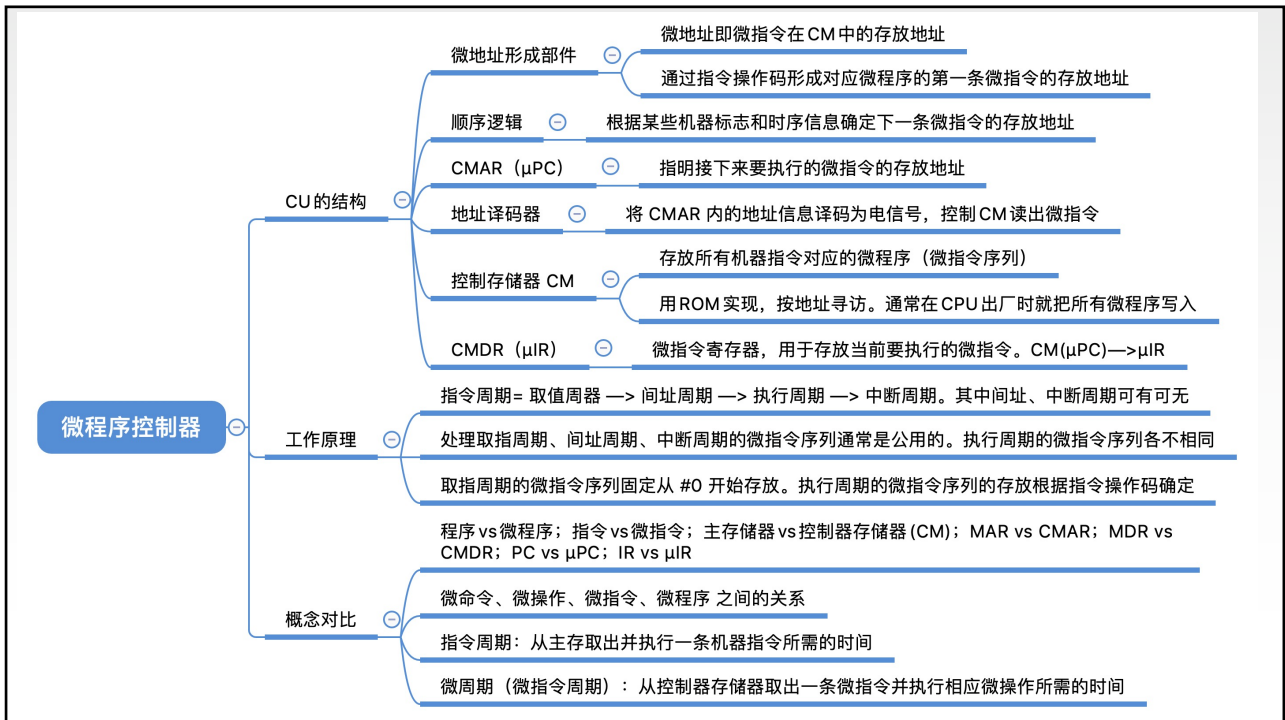
5



6



7



8

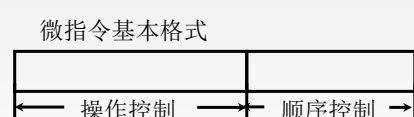
本节内容

微指令的设计

王道考研/CSKAOYAN.COM

9

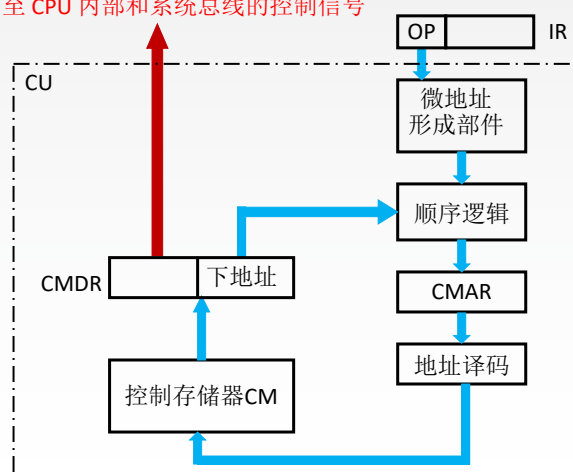
微程序控制器的工作原理



微指令的具体格式应该怎么设计？
如何根据微指令发出相应的微命令？

微命令与微操作一一对应，一个微命令对应一根输出线
有的微命令可以并行执行，因此一条微指令可以包含多个微命令

至 CPU 内部和系统总线的控制信号



王道考研/CSKAOYAN.COM

10

微指令的格式

1. 水平型微指令 一条微指令能定义多个可并行的微命令。

基本格式

如何表示一系列控制信号？



优点：微程序短，执行速度快；

缺点：微指令长，编写微程序较麻烦。

相容性微命令：可以并行完成的微命令。

互斥性微命令：不允许并行完成的微命令。

微指令1
微指令2
微指令3

水平型（胖胖的）

2. 垂直型微指令 一条微指令只能定义一个微命令，由微操作码字段规定具体功能

基本格式



优点：微指令短、简单、规整，便于编写微程序；

缺点：微程序长，执行速度慢，工作效率低。

微指令0
微指令1
微指令2
微指令3
微指令4
微指令5
微指令6

垂直型
（瘦瘦的）

3. 混合型微指令 在垂直型的基础上增加一些不太复杂的并行操作。

微指令较短，仍便于编写；微程序也不长，执行速度加快。

王道考研/CSKAOYAN.COM

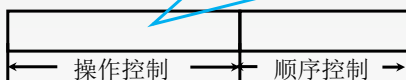
11

微指令的编码方式

1. 水平型微指令 一条微指令能定义多个可并行的微命令。

基本格式

如何表示一系列控制信号？



优点：微程序短，执行速度快；

缺点：微指令长，编写微程序较麻烦。

微指令的编码方式又称为微指令的控制方式，它是指如何对微指令的控制字段进行编码，以形成控制信号。编码的目标是在保证速度的情况下，尽量缩短微指令字长。

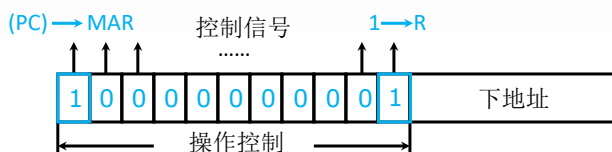
- (1) 直接编码（直接控制）方式

在微指令的操作控制字段中，每一位代表一个微操作命令

某位为“1”表示该控制信号有效

优点：简单、直观，执行速度快，操作并行性好。

缺点：微指令字长过长， n 个微命令就要求微指令的操作字段有 n 位，造成控存容量极大。



王道考研/CSKAOYAN.COM

12

微指令的编码方式

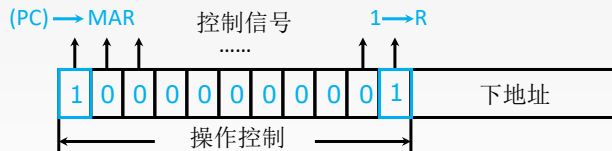
(1) 直接编码（直接控制）方式

在微指令的操作控制字段中，**每一位代表一个微操作命令**

某位为“1”表示该控制信号有效

优点：简单、直观，执行速度快，操作并行性好。

缺点：微指令字长过长， n 个微命令就要求微指令的操作字段有 n 位，造成控存容量极大。

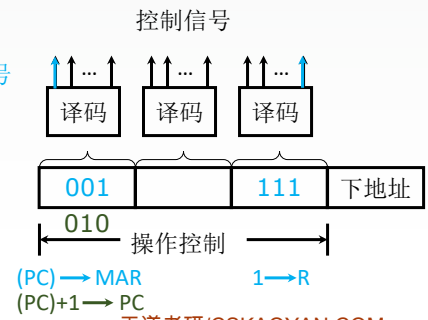


(2) 字段直接编码方式

将微指令的控制字段分成若干“段”，**每段经译码后发出控制信号**

微命令字段分段的原则：

- ① **互斥性**微命令分在**同一段内**，**相容性**微命令分在**不同段内**。
- ② **每个小段中包含的信息位不能太多**，否则将增加译码线路的复杂性和译码时间。
- ③ 一般**每个小段还要留出一个状态**，表示本字段不发出任何微命令。因此，当某字段的长度为3位时，最多只能表示7个互斥的微命令，**通常用000表示不操作**。



王道考研/CSKAOYAN.COM

13

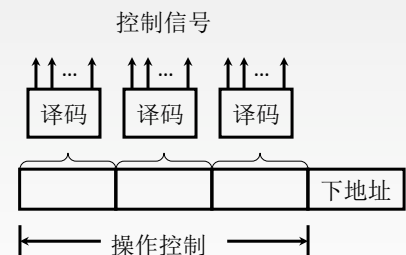
例题：字段直接编码方式

(2) 字段直接编码方式

将微指令的控制字段分成若干“段”，**每段经译码后发出控制信号**

微命令字段分段的原则：

- ① **互斥性**微命令分在**同一段内**，**相容性**微命令分在**不同段内**。
- ② **每个小段中包含的信息位不能太多**，否则将增加译码线路的复杂性和译码时间。
- ③ 一般**每个小段还要留出一个状态**，表示本字段不发出任何微命令。因此，当某字段的长度为3位时，最多只能表示7个互斥的微命令，**通常用000表示不操作**。



某计算机的控制器采用微程序控制方式，微指令中的操作控制字段采用**字段直接编码法**，共有33个微命令，构成5个互斥类，分别包含7、3、12、5和6个微命令，则操作控制字段至少有多少位？

第1个互斥类有7个微命令，**要留出1个状态表示不操作**，所以需要表示8种不同的状态，故需要3个二进制位。

故操作控制字段的总位数为

$$3+2+4+3+3 = 15 \text{ 位}$$

以此类推，后面4个互斥类各需要表示4、13、6、7种不同的状态，分别对应2、4、3、3个二进制位。

Tips: 若采用**直接编码**方式，则控制字段需要33位

王道考研/CSKAOYAN.COM

14

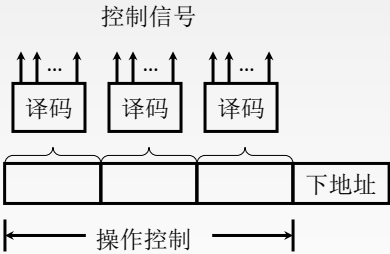
微指令的编码方式

(2) 字段直接编码方式

将微指令的控制字段分成若干“段”，每段经译码后发出控制信号

微命令字段分段的原则：

- ① 互斥性微命令分在同一段内，相容性微命令分在不同段内。
- ② 每个小段中包含的信息位不能太多，否则将增加译码线路的复杂性和译码时间。
- ③ 一般每个小段还要留出一个状态，表示本字段不发出任何微命令。因此，当某字段的长度为3位时，最多只能表示7个互斥的微命令，通常用000表示不操作。

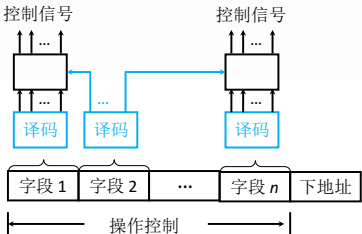


优点：可以缩短微指令字长。

缺点：要通过译码电路后再发出微命令，因此比直接编码方式慢。

(3) 字段间接编码方式

一个字段的某些微命令需由另一个字段中的某些微命令来解释，由于不是靠字段直接译码发出的微命令，故称为字段间接编码，又称隐式编码。



优点：可进一步缩短微指令字长。

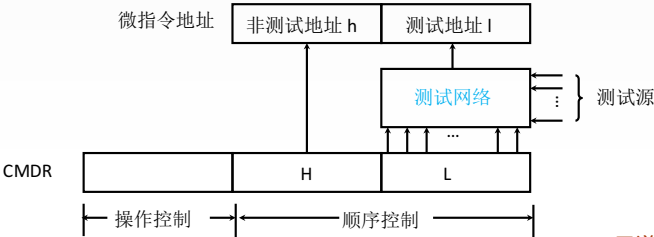
缺点：削弱了微指令的并行控制能力，故通常作为字段直接编码方式的一种辅助手段。

微指令的地址形成方式

- 1. 微指令的 下地址字段 指出 微指令格式中设置一个下地址字段，由微指令的下地址字段直接指出后继微指令的地址，这种方式又称为**断定方式**。
- 2. 根据机器指令的 操作码 形成 当机器指令取至指令寄存器后，微指令的地址由操作码经微地址形成部件形成。
- 3. 增量**计数器法** $(CMAR) + 1 \rightarrow CMAR$
- 4. 分支转移 转移方式：指明判别条件；转移地址：指明转移成功后的去向。



5. 通过测试网络



微指令的地址形成方式

1. 微指令的 **下地址字段** 指出 微指令格式中设置一个下地址字段，由微指令的下地址字段直接指出后继微指令的地址，这种方式又称为**断定方式**。
2. 根据机器指令的 **操作码** 形成 当机器指令取至指令寄存器后，微指令的地址由操作码经微地址形成部件形成。
3. 增量**计数器法** $(CMAR) + 1 \rightarrow CMAR$
4. 分支转移 转移方式：指明判别条件；转移地址：指明转移成功后的去向。

操作控制字段	转移方式	转移地址
--------	------	------

5. 通过测试网络
6. 由硬件产生微程序入口地址
第一条微指令地址 由专门 **硬件** 产生（用专门的硬件记录取指周期微程序首地址）
 中断周期 由 **硬件** 产生 **中断周期微程序首地址**（用专门的硬件记录）

王道考研/CSKAOYAN.COM

17

例题-断定方式

1. 微指令的 **下地址字段** 指出 微指令格式中设置一个下地址字段，由微指令的下地址字段直接指出后继微指令的地址，这种方式又称为**断定方式**。

某计算机采用微程序控制器，共有32条指令，公共的取指令微程序包含2条微指令，各指令对应的微程序平均由4条微指令组成，采用**断定法**（下地址字段法）确定下条微指令地址，则微指令中下地址字段的位数至少是多少位？

王道考研/CSKAOYAN.COM

18

例题-断定方式

微指令存放地址

微指令存放地址	微指令操作码	下地址
0		1
1		2
2		3
3	⋮ 转执行周期	
4		
5		
6		
7		
8	⋮ 转取指周期	
9		
10		
11		
12		
13		14
14		15
15		0
16		17
17		18
18		0

取指周期 微程序
间址周期 微程序
中断周期 微程序

对应 LDA 指令的执行周期微程序
对应 STA 指令的执行周期微程序

某计算机采用微程序控制器，共有32条指令，公共的取指令微程序包含2条微指令，各指令对应的微程序平均由4条微指令组成，采用**断定法**（下地址字段法）确定下条微指令地址，则微指令中下地址字段的位数至少是多少位？

总共需要存储多少条微指令？
 $32 \times 4 + 2 = 130$ 条

标注出130个不同的位置至少需要多少个二进制位？
 $2^7 = 128, 2^8 = 256$

下地址字段的位数至少是8位

王道考研/CSKAOYAN.COM

19

知识回顾

微指令的设计

- 微指令格式
 - 水平型微指令：每条微指令能定义多个可并行的微命令
 - 垂直型微指令：每条微指令只能定义一个微命令，由微操作码指明
 - 混合型微指令：在垂直型微指令的基础上加上一些简单的并行操作
- 水平型微指令的编码方式
 - 直接编码（直接控制）：控制码的每个bit对应一个微命令，微指令执行速度最快
 - 将互斥性的微命令分在同一个段内，相容的分在不同的段
 - 字段直接编码：每个段留出一个状态表示“不操作”
 - 微指令操作码需要经过译码电路处理，因此执行速度更慢
 - 字段间接编码：一个字段的微命令需要用另一个字段的微命令解释
 - 可能需要多级译码电路处理，执行速度最慢
- 下一条微指令地址的形成方式
 - 断定法（下地址法）：根据当前执行的微指令下地址找到下一条微指令
 - 计数器法： $\mu PC + 1$ 顺序找到下一条微指令
 - 根据指令操作码确定执行周期微程序首地址
 - 由专门的硬件指明取指/中断周期的微程序首地址

王道考研/CSKAOYAN.COM

20

本节内容

微程序控制单元的设计

王道考研/CSKAOYAN.COM

21

微程序控制单元的设计

设计步骤:

1. 分析每个阶段的微操作序列
2. 写出对应机器指令的微操作命令及节拍安排
3. 确定微指令格式
4. 编写微指令码点

取指周期-硬布线控制器的节拍安排

T_0 $PC \rightarrow MAR$
 T_0 $1 \rightarrow R$
 T_1 $M(MAR) \rightarrow MDR$
 T_1 $(PC) + 1 \rightarrow PC$
 T_2 $MDR \rightarrow IR$
 T_2 $OP(IR) \rightarrow ID$

取指周期-微程序控制器的节拍安排

T_0 $PC \rightarrow MAR$
 T_0 $1 \rightarrow R$
 T_1 $M(MAR) \rightarrow MDR$
 T_1 $(PC) + 1 \rightarrow PC$
 T_2 $MDR \rightarrow IR$
 T_2 $OP(IR) \rightarrow \text{微地址形成部件}$

3 条微指令

王道考研/CSKAOYAN.COM

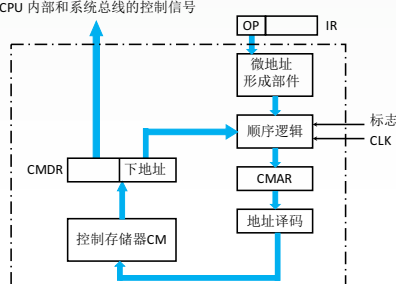
22

微程序控制单元的设计

取指周期-硬布线控制器的节拍安排

T_0 $PC \rightarrow MAR$
 T_0 $1 \rightarrow R$
 T_1 $M(MAR) \rightarrow MDR$
 T_1 $(PC) + 1 \rightarrow PC$
 T_2 $MDR \rightarrow IR$
 T_2 $OP(IR) \rightarrow ID$

至 CPU 内部和系统总线的控制信号



取指周期-微程序控制器的节拍安排

T_0 $PC \rightarrow MAR$
 T_0 $1 \rightarrow R$
 T_1 $M(MAR) \rightarrow MDR$
 T_1 $(PC) + 1 \rightarrow PC$
 T_2 $MDR \rightarrow IR$
 T_2 $OP(IR) \rightarrow \text{微地址形成部件}$

还需考虑如何读出这3条微指令，以及如何转入下一个机器周期

$Ad(CMDR) \rightarrow CMAR$

$OP(IR) \rightarrow \text{微地址形成部件} \rightarrow CMAR$

取指周期的第一条微指令地址由硬件自动给出

微指令a

用微指令a的下地址表示b的地址

微指令b

微指令c

用当前微指令的下地址表示找到下一条微指令

王道考研/CSKAOYAN.COM

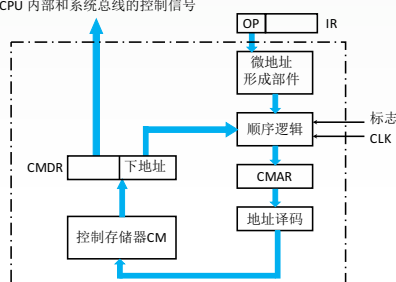
23

微程序控制单元的设计

取指周期-硬布线控制器的节拍安排

T_0 $PC \rightarrow MAR$
 T_0 $1 \rightarrow R$
 T_1 $M(MAR) \rightarrow MDR$
 T_1 $(PC) + 1 \rightarrow PC$
 T_2 $MDR \rightarrow IR$
 T_2 $OP(IR) \rightarrow ID$

至 CPU 内部和系统总线的控制信号



取指周期-微程序控制器的节拍安排

T_0 $PC \rightarrow MAR$
 T_0 $1 \rightarrow R$
 T_1 $Ad(CMDR) \rightarrow CMAR$
 T_2 $M(MAR) \rightarrow MDR$
 T_2 $(PC) + 1 \rightarrow PC$
 T_3 $Ad(CMDR) \rightarrow CMAR$
 T_4 $MDR \rightarrow IR$
 T_4 $OP(IR) \rightarrow \text{微地址形成部件}$
 T_5 $\text{微地址形成部件} \rightarrow CMAR$

微指令a

需要用 T_1 节拍确定下一条微指令的地址

微指令b

需要用 T_3 节拍确定下一条微指令的地址

微指令c

根据指令操作码确定其执行周期微指令序列的首地址

显然，微程序控制器的速度比硬布线控制器更慢

王道考研/CSKAOYAN.COM

24

微程序控制单元的设计

取指周期-硬布线控制器的节拍安排

T_0 $PC \rightarrow MAR$

T_0 $1 \rightarrow R$

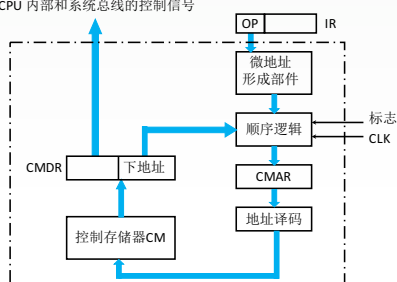
T_1 $M(MAR) \rightarrow MDR$

T_1 $(PC) + 1 \rightarrow PC$

T_2 $MDR \rightarrow IR$

T_2 $OP(IR) \rightarrow ID$

至 CPU 内部和系统总线的控制信号



取指周期-微程序控制器的节拍安排

T_0 $PC \rightarrow MAR$

T_0 $1 \rightarrow R$

微指令a

T_1 $Ad(CMDR) \rightarrow CMAR$

T_2 $M(MAR) \rightarrow MDR$

T_2 $(PC) + 1 \rightarrow PC$

微指令b

T_3 $Ad(CMDR) \rightarrow CMAR$

T_4 $MDR \rightarrow IR$

微指令c

T_5 $OP(IR) \rightarrow \text{微地址形成部件} \rightarrow CMAR$

取指周期最后一条微指令完成后，用一个特殊的微操作确定执行周期的微程序首地址

显然，微程序控制器的速度比硬布线控制器更慢

王道考研/CSKAOYAN.COM

25

微程序控制单元的设计

设计步骤：

1. 分析每个阶段的微操作序列
2. 写出对应机器指令的微操作命令及节拍安排
 - (1) 写出每个周期所需要的微操作(参照硬布线)
 - (2) 补充微程序控制器特有的微操作：

每条微指令结束之后都需要进行

每条微指令结束之后都需要进行

a. 取指周期：

$Ad(CMDR) \rightarrow CMAR$

$OP(IR) \rightarrow \text{微地址形成部件} \rightarrow CMAR$

b. 执行周期：

$Ad(CMDR) \rightarrow CMAR$

取指周期的最后一条微指令完成后，要根据指令操作码确定其执行周期的微程序首地址

3. 确定微指令格式
4. 编写微指令码点

王道考研/CSKAOYAN.COM

26

微程序控制单元的设计

设计步骤:

1. 分析每个阶段的微操作序列
2. 写出对应机器指令的微操作命令及节拍安排
 - (1) 写出每个周期所需要的微操作(参照硬布线)
 - (2) 补充微程序控制器特有的微操作:
 - a. 取指周期:
 - Ad (CMDR) \rightarrow CMAR
 - OP (IR) \rightarrow CMAR
 - b. 执行周期:
 - Ad(CMDR) \rightarrow CMAR
3. 确定微指令格式

根据微操作个数决定采用何种编码方式, 以确定微指令的操作控制字段的位数。
 根据CM中存储的微指令总数, 确定微指令的顺序控制字段的位数。
 最后按操作控制字段位数和顺序控制字段位数就可确定微指令字长。
4. 编写微指令码点

王道考研/CSKAOYAN.COM

27

微程序控制单元的设计

设计步骤:

1. 分析每个阶段的微操作序列
2. 写出对应机器指令的微操作命令及节拍安排
 - (1) 写出每个周期所需要的微操作(参照硬布线)
 - (2) 补充微程序控制器特有的微操作:
 - a. 取指周期:
 - Ad (CMDR) \rightarrow CMAR
 - OP (IR) \rightarrow CMAR
 - b. 执行周期:
 - Ad(CMDR) \rightarrow CMAR
3. 确定微指令格式

根据微操作个数决定采用何种编码方式, 以确定微指令的操作控制字段的位数。
 根据CM中存储的微指令总数, 确定微指令的顺序控制字段的位数。
 最后按操作控制字段位数和顺序控制字段位数就可确定微指令字长。
4. 编写微指令码点

根据操作控制字段每一位代表的微操作命令, 编写每一条微指令的码点。

王道考研/CSKAOYAN.COM

28

微程序设计分类

1. 静态微程序设计和动态微程序设计

静态 微程序无需改变，采用 **ROM**

动态 通过 **改变微指令** 和 **微程序** 改变机器指令
有利于仿真，采用 **EPROM**

2. 毫微程序设计

毫微程序设计的基本概念

微程序设计 用 **微程序解释机器指令**

毫微程序设计 用 **毫微程序解释微程序**

毫微指令与微指令 的关系好比 **微指令与机器指令** 的关系

王道考研/CSKAOYAN.COM

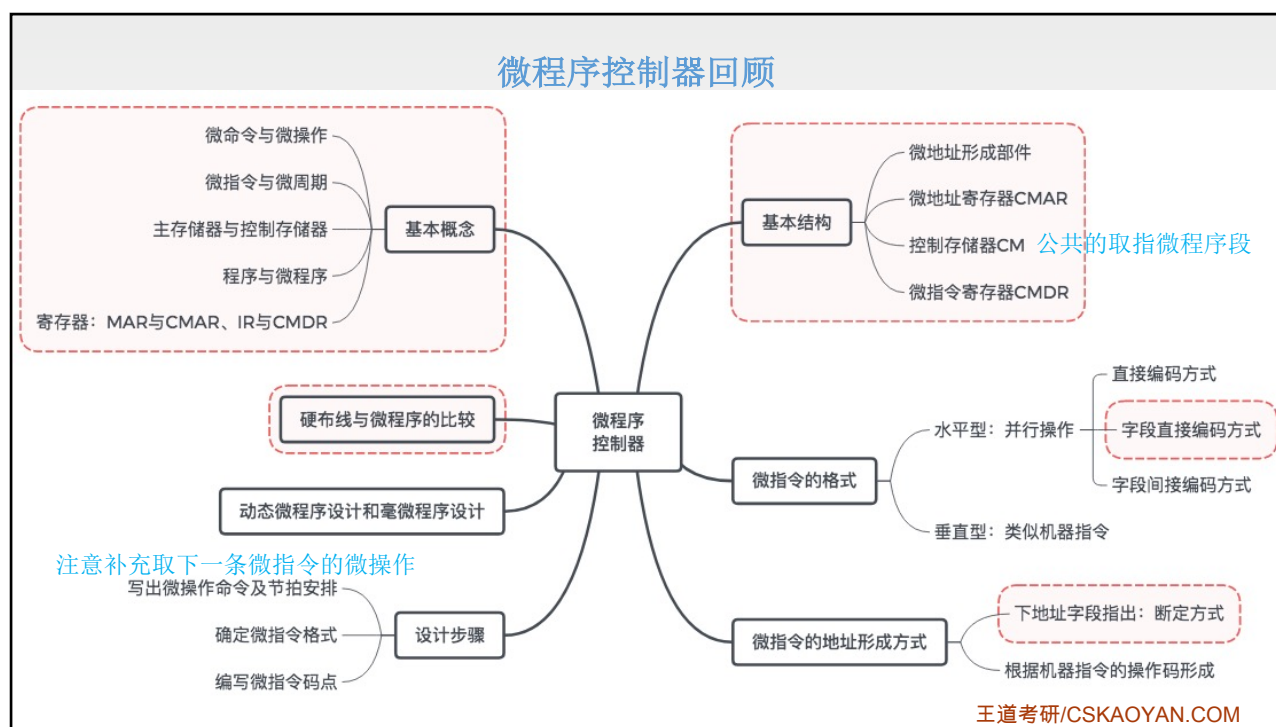
29

硬布线与微程序的比较

类 别 对比项目	微程序控制器	硬布线控制器
工作原理	微操作控制信号以微程序的形式存放在控制存储器中，执行指令时读出即可	微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生
执行速度	慢	快
规整性	较规整	烦琐、不规整
应用场合	CISC CPU	RISC CPU
易扩充性	易扩充修改	困难

王道考研/CSKAOYAN.COM

30



31



32